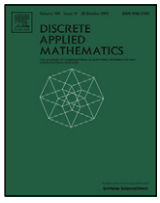
Discrete Applied Mathematics 160 (2012) 1901–1913

Contents lists available at SciVerse ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam



Online maximum k-coverage

G. Ausiello [a], N. Boria [b], A. Giannakos [b], G. Lucarelli [b,*], V.Th. Paschos [b,c]

[a] *Dipartimento di Informatica e Sistemistica, Università degli Studi di Roma "La Sapienza", Italy*

[b] *Paris Sciences et Lettres Research University, Université Paris-Dauphine, LAMSADE, CNRS UMR 7243, France*

[c] *Institut Universitaire de France, France*

ARTICLE INFO

Article history:
Received 29 June 2011
Received in revised form 8 February 2012
Accepted 9 April 2012
Available online 9 May 2012

Keywords:
Maximum k-coverage
Competitive ratio
Negative results
Graphs

ABSTRACT

We study an online model for the maximum k-vertex-coverage problem, in which, given a graph $G = (V, E)$ and an integer k, we seek a subset $A \subseteq V$ such that $|A| = k$ and the number of edges covered by A is maximized. In our model, at each step i, a new vertex v_i is released, and we have to decide whether we will keep it or discard it. At any time of the process, only k vertices can be kept in memory; if at some point the current solution already contains k vertices, any inclusion of a new vertex in the solution must entail the definite deletion of another vertex of the current solution (a vertex not kept when released is definitely deleted). We propose algorithms for several natural classes of graphs (mainly regular and bipartite), improving on an easy $\frac{1}{2}$-competitive ratio. We next settle a set version of the problem, called the maximum k-(set)-coverage problem. For this problem, we present an algorithm that improves upon former results for the same model for small and moderate values of k.

1. Introduction

In the *maximum k-vertex-coverage* (denoted by MAX k-VERTEX COVERAGE) problem, we are given a graph $G = (V, E)$ ($|V| = n$, $|E| = m$) and an integer k, and we seek a subset $A \subseteq V$ such that $|A| = k$ and the number of edges covered by A is maximized. The MAX k-VERTEX COVERAGE problem is NP-hard, since otherwise the optimal solution for the vertex cover problem could be found in polynomial time: for each k, $1 \leq k \leq n$, run the algorithm for the MAX k-VERTEX COVERAGE problem and stop when all elements are covered.

In this paper, we consider the following online model for this problem: at each step i, a new vertex v_i with its adjacent edges is released, and we have to decide whether we will include v_i in the solution or discard it. At any time of the process, only k vertices can be kept in memory, so, if at some point the current solution already contains k vertices, any inclusion of any new vertex in the solution must be compensated with the definite deletion of one vertex of the current solution. Of course, a vertex that is not kept when it is released is also definitely deleted. To our knowledge, no online model for the MAX k-VERTEX COVERAGE problem has been studied until now.

A generalization of the MAX k-VERTEX COVERAGE problem is the *maximum k-(set)-coverage* (denoted by MAX k-SET COVERAGE) problem, in which, given a universe of elements $\mathcal{E} = \{e_1, e_2, \ldots, e_m\}$, a collection $S = \{S_1, S_2, \ldots, S_n\}$ of subsets of $\mathcal{E}$, and an integer $k \leq n$, we seek a subcollection $A = \{A_1, A_2, \ldots, A_{|A|}\} \subseteq S$ such that $|A| = k$ and the number of elements

* Corresponding author. Fax: +33 1 44 05 40 91.
E-mail addresses: ausiello@dis.uniroma1.it (G. Ausiello), boria@lamsade.dauphine.fr (N. Boria), giannako@lamsade.dauphine.fr (A. Giannakos), lucarelli@lamsade.dauphine.fr, glucarelli@gmail.com (G. Lucarelli), paschos@lamsade.dauphine.fr (V.Th. Paschos).

0166-218X/$ – see front matter
doi:10.1016/j.dam.2012.04.005

of $\mathcal{E}$ covered by A is maximized. The online model for the MAX k-SET COVERAGE problem is the same as for the MAX k-VERTEX COVERAGE.

Clearly, the MAX k-VERTEX COVERAGE problem is a special case of the MAX k-SET COVERAGE problem, in which (i) each element belongs to exactly two sets and (ii) the intersection of any two sets of S has size at most one, since multiple edges are not permitted.

The weighted generalization of the MAX k-SET COVERAGE problem, denoted by WEIGHTED MAX k-SET COVERAGE, has been also studied in the literature. In this problem, each element $e_i \in \mathcal{E}$ has a non-negative weight $w(e_i)$, and the goal is to maximize the total weight of the elements covered by k sets.

The analogous online model for the WEIGHTED MAX k-SET COVERAGE problem, where at each step i a set $S_i \in S$ together with its elements is released and only k such sets can be kept in memory, has been studied in [15], where an algorithm of competitive ratio $\frac{1}{4}$ is given. The authors in their so-called *set-streaming model* assume that the universe of the instance is known *a priori*. Nevertheless, they do not use this information in the proposed algorithm.

In the classic offline setting, the MAX k-SET COVERAGE problem is known to be non-approximable within a factor $1 - \frac{1}{e}$ [7]. On the other hand, even for the weighted version of the problem, an approximation algorithm of ratio $1 - \left(1 - \frac{1}{k}\right)^k$ is known [12]. This ratio tends to $1 - \frac{1}{e}$ as k increases, closing in this way the approximability question for the problem.

In [1], the inverse problem (i.e., the hitting set version of the MAX k-SET COVERAGE problem), also called the *maximum coverage problem*, has been studied: given a universe of elements $\mathcal{E} = \{e_1, e_2, \ldots, e_m\}$, a collection of subsets of $\mathcal{E}$, $S = \{S_1, S_2, \ldots, S_n\}$, a non-negative weight $w(S_i)$ for each $S_i \in S$, and an integer k, a set $B \subseteq \mathcal{E}$ is sought such that $|B| = k$ and the total weight of the sets in S that intersect with B is maximized. It is easy to see that this version is equivalent to the WEIGHTED MAX k-SET COVERAGE problem modulo the interchange of the roles between a set system and the universe of elements. An algorithm of approximation ratio $1 - \left(1 - \frac{1}{p}\right)^p$ is presented in [1] for this problem, where p is the cardinality of the largest set in S. In the case where each set has cardinality equal to two, then this problem coincides with the MAX k-VERTEX COVERAGE problem; hence a $\frac{3}{4}$ approximation ratio is implied by the algorithm in [1]. Using semidefinite programming, a 0.8-approximation algorithm has been proposed in [8] for the case where $k \geq n/2$, while an algorithm that improves any known ratio for some values of k and n is presented in [11].

In [4], an algorithm is presented about another online model for the MAX k-SET COVERAGE problem: the sets are known in advance while the elements arrive online. The list of sets that contain each element becomes known when this element arrives.

A similar setting for the online set cover problem has been also studied in [2,5]. The difference is that not all elements will finally arrive, without knowing *a priori* which of them will do. Finally, in the model studied in [3] for the set cover problem, when an element arrives, some information about the list of sets that contain it appears too (e.g., the maximum cardinality set that contains it or the set from its list that covers the larger number of not yet released elements, ...). We also mention here another variant of the online set cover problem analyzed in [10].

In this paper, we study the online model described above for both the MAX k-VERTEX COVERAGE problem and the MAX k-SET COVERAGE problem. In Section 2, we prove several negative results on the competitiveness of any algorithm for the model handled for both problems. In Section 3, we present algorithms for regular graphs, regular bipartite graphs, trees, and chains, achieving non-trivial competitive ratios, improving upon an easy $\frac{1}{2}$ competitiveness result holding for any graph. Finally, in Section 4, the MAX k-SET COVERAGE problem is handled. For this problem, we present an algorithm that generalizes the $\frac{1}{4}$-competitive algorithm presented in [15] and improves upon former results for the same model for small and moderate values of k.

The following notation will be used in what follows. It is based upon the definition of the MAX k-VERTEX COVERAGE problem and is easily extendable to the MAX k-SET COVERAGE problem.

For any $A \subseteq V$, we denote by $E(A)$ the set of edges covered by A and by $m(A) = |E(A)|$ the number of these edges. Let $SOL = m(A)$ be the number of edges covered by our algorithms. Moreover, we denote by $A^* \subseteq V$ an optimal subset of vertices and by $OPT = m(A^*)$ the number of edges covered by an optimal solution. The degree of a vertex $v \in V$ is denoted by $d(v)$, while the maximum degree (or the degree when it is regular) of the input graph $G = (V, E)$ is denoted by Δ. Dealing with MAX k-SET COVERAGE, Δ denotes the cardinality of a set of maximum size, that is, $\Delta = \max\{|S_i|: 1 \leq i \leq n\}$. For a subset $A \subseteq V$ and a vertex $v_i \in A$, we call *public* the edges incident to v_i and to another vertex in A and *private* the edges of v_i that are covered just by v_i in A. Finally, as is common in the online setting, the quality of an algorithm is measured by means of the so-called *competitive ratio*, representing the ratio of the value of the solution computed by the algorithm over the optimal value of the whole instance, i.e., the value of an optimal (offline) solution of the final instance.

2. Negative results

In this section, we give negative results for the online MAX k-VERTEX COVERAGE problem and their corresponding adaptations for the MAX k-SET COVERAGE problem. We start with a negative result for the case where we do not allow any "swaps", i.e., where the replacement of a vertex or set that belongs to the current solution by the newly released vertex or set is not permitted.

Proposition 1. *Any deterministic online algorithm that does not allow swaps cannot achieve a competitive ratio better than*

- $O\left(\frac{1}{(n-1)^{1/(k+1)}}\right)$, *for the* MAX k-VERTEX COVERAGE *problem,*
- $O\left(\frac{1}{m^{1/(k+1)}}\right)$, *for the* MAX k-SET COVERAGE *problem.*

Proof. For the MAX k-VERTEX COVERAGE problem, let $k \ll n$, and consider the following scenario. In step i, $1 \leq i \leq k$, the central vertex v_i of a star with $d(v_i) = (n-1)^{i/(k+1)}$ is released. If the algorithm rejects v_i, then the remaining $\sum_{j=1}^{i}(n-1)^{j/(k+1)}$ vertices of the i stars plus $n - i - \sum_{j=1}^{i}(n-1)^{j/(k+1)}$ singleton vertices are released. If the algorithm selects v_i, then vertex v_{i+1}, with

$$d(v_{i+1}) = (n-1)^{(i+1)/(k+1)},$$

is released. If after the kth vertex the algorithm has selected all the k released vertices, then a new vertex v_{k+1} with degree $d(v_{k+1}) = n-1$ is released; finally, the remaining vertices of the stars and $n - k - \sum_{j=1}^{k}(n-1)^{j/(k+1)}$ singleton vertices are released.

If after step i the algorithm has rejected v_i, then only vertices of degree at most one are released. Hence, the algorithm covers $k-(i-1)+\sum_{j=1}^{i-1}(n-1)^{j/(k+1)}$ edges, while the optimum solution covers $k-i+\sum_{j=1}^{i}(n-1)^{j/(k+1)}$ edges. Thus

$$\frac{SOL}{OPT} = \frac{k-(i-1)+\sum_{j=1}^{i-1}(n-1)^{j/(k+1)}}{k-i+\sum_{j=1}^{i}(n-1)^{j/(k+1)}} = \frac{k-(i-1)+\frac{(n-1)^{i/(k+1)}-(n-1)^{1/(k+1)}}{(n-1)^{1/(k+1)}-1}}{k-i+\frac{(n-1)^{(i+1)/(k+1)}-(n-1)^{1/(k+1)}}{(n-1)^{1/(k+1)}-1}} = O\left(\frac{1}{(n-1)^{1/(k+1)}}\right).$$

If the algorithm has selected all the k first released vertices, then it covers exactly $\sum_{j=1}^{k}(n-1)^{j/(k+1)}$ elements, while the optimum solution (that includes v_{k+1}, which is never selected by the online algorithm) covers $n-1$ elements. Hence

$$\frac{SOL}{OPT} = \frac{\sum_{j=1}^{k}(n-1)^{j/(k+1)}}{n-1} = \frac{\frac{(n-1)^{(k+1)/(k+1)}-(n-1)^{1/(k+1)}}{(n-1)^{1/(k+1)}-1}}{n-1} = O\left(\frac{1}{(n-1)^{1/(k+1)}}\right),$$

which concludes the proof.

For the MAX k-SET COVERAGE problem, the proof is similar. In this case, in phase i, if all previously released sets are selected by the algorithm, then a set of cardinality $m^{i/(k+1)}$ is released. □

The next negative result for the MAX k-VERTEX COVERAGE problem fits the model addressed in the paper (swaps are allowed).

Proposition 2. *Any deterministic online algorithm cannot achieve a competitive ratio better than* $\frac{2k}{3k-2} \simeq \frac{2}{3}$ *for the* MAX k-VERTEX COVERAGE *problem.*

Proof. Assume that $2k-1$ vertices, $v_1^1, v_2^1, \ldots, v_{2k-1}^1$, of degree one and $2k-1$ vertices, $v_1^2, v_2^2, \ldots, v_{2k-1}^2$, of degree two are released such that $(v_i^1, v_i^2) \in E$, $1 \leq i \leq 2k-1$, and that the algorithm selects $k' \leq k$ of them.

Without loss of generality, let $v_1^2, v_2^2, \ldots, v_{k'}^2$ be the vertices selected by the algorithm (step (1) in Fig. 1). Next, vertex v_3 of degree k' is released, where $(v_i^2, v_3) \in E$, $1 \leq i \leq k'$. The solution of the algorithm at this time is $2k'$, while the inclusion or not of v_3 does not play any role in this value. Finally, $2k-1-k'$ vertices, $v_{k'+1}^3, v_{k'+2}^3, \ldots, v_{2k-1}^3$, of degree one are released, such that $(v_i^2, v_i^3) \in E$, $k'+1 \leq i \leq 2k-1$. In this last phase, the algorithm can increase its solution by at most $k-k'$ more edges (step (2) in Fig. 1). Hence, the final solution of the algorithm is at most $k+k'$. The optimum solution consists of the vertices $v_{k+1}^2, v_{k+2}^2, \ldots, v_{2k-1}^2, v_3$, and hence is of cardinality $2(k-1)+k'$. Therefore, $\frac{SOL}{OPT} = \frac{k+k'}{2(k-1)+k'} \leq \frac{2k}{3k-2}$. □

An analogous result can be proved for the MAX k-SET COVERAGE problem. Recall that for the offline version of the MAX k-SET COVERAGE problem an $1-\frac{1}{e} \simeq 0.63$-inapproximability result is known [7].

Proposition 3. *Any deterministic online algorithm cannot achieve a competitive ratio better than* $\frac{k+2\sqrt{k}+1}{2k+2\sqrt{k}+1} \simeq \frac{1}{2}$ *for the* MAX k-SET COVERAGE *problem even in the case where all sets have the same cardinality.*

Proof. An *r-sunflower* is a set system of regular sets of size Δ with a common intersection of size r; the sets of a sunflower are called *petals*.

Consider the following scenario. The adversary starts by sending $\frac{\Delta(p-1)}{p}$-sunflower petals, where $\frac{\Delta}{p} \in \mathbb{N}$, while the algorithm keeps k' of them; it continues so until the first time τ where there are $k - \lfloor \frac{k'}{p} \rfloor$ rejected sets. Notice that this will be always the case for some $\tau \leq \lceil \frac{k(2p-1)}{p} \rceil$.

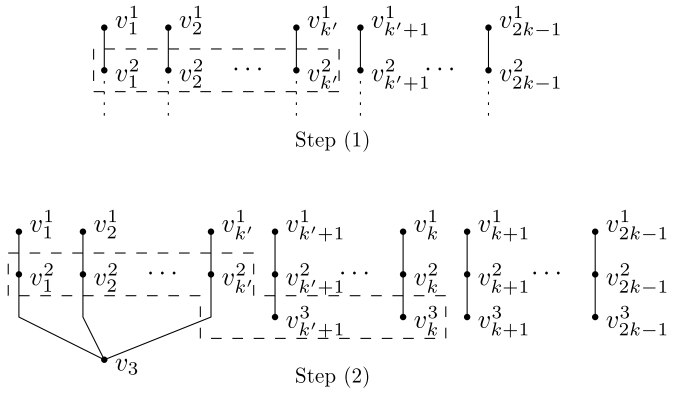


Fig. 1. An illustration of the counterexample of Proposition 2.

Then the adversary starts sending disjoint sets, each one matching the private parts of p petals in the solution, until the maximum number of private parts have been matched.

The solution of the algorithm will cover at most $\frac{(k'+p-1)}{p}\Delta$ elements, while the optimum will cover at least $\lfloor\frac{k'}{p}\rfloor\Delta$ elements by the matching sets plus $\lceil\frac{k-\frac{k'}{p}+p-1}{p}\rceil\Delta$ elements by rejected petals. Thus, the ratio is bounded above by $\frac{k'+p-1}{k+(p-1)\frac{k'}{p}+p-1}$, where $0 \leq k' \leq k$, which is less than or equal to the simplified expression $\frac{pk+p(p-1)}{(2p-1)k+p(p-1)}$. This expression is minimized when $p = \sqrt{k}+1$, that is,

$$\frac{(\sqrt{k}+1)k+(\sqrt{k}+1)\sqrt{k}}{(2(\sqrt{k}+1)-1)k+(\sqrt{k}+1)\sqrt{k}} = \frac{k+2\sqrt{k}+1}{2k+2\sqrt{k}+1},$$

which for k large enough tends asymptotically to $\frac{1}{2}$. □

3. Maximum *k*-vertex-coverage

In this section, we deal with the online MAX k-VERTEX COVERAGE problem. Note, first, that there exists an easy $\frac{1}{2}$-competitive ratio for this problem. In fact, consider selecting k vertices of largest degrees. In an optimum solution, all the edges are, at best, covered once, while, in the solution created by this greedy algorithm, all the edges are, at worst, covered twice. Since the algorithm selects the vertices with the highest degrees of the graph, the $\frac{1}{2}$-competitive ratio is immediately concluded.

Proposition 4. *There is a $\frac{1}{2}$-competitive ratio for the online* MAX k-VERTEX COVERAGE *problem.*

In the rest of this section, we improve the $\frac{1}{2}$-competitive ratio for several classes of graphs. But first we give an easy upper bound for the number of elements covered by any solution, which will be used later. Its proof is straightforward.

Remark 1. $OPT \leq k\Delta$.

3.1. Regular graphs

The MAX k-VERTEX COVERAGE problem is NP-complete in regular graphs, a result that directly occurs by the NP-completeness of the minimum vertex cover problem in cubic graphs [9].

The following preliminary result, which will be used later, holds for any algorithm for the MAX k-VERTEX COVERAGE problem in regular graphs.

Proposition 5. *Any deterministic online algorithm achieves a $\frac{k}{n}$-competitive ratio for the* MAX k-VERTEX COVERAGE *problem on regular graphs.*

Proof. An optimum solution covers at most all the edges of the graph (recall that $|E| = m$); that is, $OPT \leq m = \frac{n\Delta}{2}$. On the other hand, any solution (included the optimum one) covers at most $k\Delta$ edges and at least $\frac{k\Delta}{2}$ edges, some of them being possibly covered twice; that is, $SOL \geq \frac{k\Delta}{2}$. We therefore get $\frac{SOL}{OPT} \geq \frac{k}{n}$. □

Let us note that the result of Proposition 5 for the MAX k-VERTEX COVERAGE problem also holds for general graphs in the offline setting [11].

We now present an algorithm for the MAX k-VERTEX COVERAGE problem in regular graphs. Our algorithm depends on a parameter x which indicates the improvement on the current solution that a new vertex should entail, in order to be selected for inclusion in the solution. In other words, we replace a vertex of the current solution by the released one only if the solution increases by at least $\lceil \frac{\Delta}{x} \rceil$ edges.

ALGORITHM Mk VC-R(x)
1: $A = \emptyset$; (the solution of the algorithm)
2: $B = \emptyset$; (the set of vertices that increase the solution by at least $\lceil \frac{\Delta}{x} \rceil$)
3: **for** each released vertex v **do**
4: **if** $|A| < k$ **then**
5: $A = A \cup \{v\}$;
6: **if** $|E(\{v\}) \setminus E(B)| \geq \lceil \frac{\Delta}{x} \rceil$ **then**
7: $B = B \cup \{v\}$;
8: **else if** $|B| < k$ **and** $|E(\{v\}) \setminus E(B)| \geq \lceil \frac{\Delta}{x} \rceil$ **then**
9: select a vertex $u \in A \setminus B$;
10: $A = A \cup \{v\} \setminus \{u\}$; $B = B \cup \{v\}$;
11: **return** A;

As we will see in what follows, the best value for x is $x = \frac{n+2k+\sqrt{4k^2+n^2}}{2n}$, leading to the following theorem.

Theorem 1. ALGORITHM Mk VC-R *achieves a 0.55-competitive ratio.*

Proof. Note that $B \subseteq A$ consists of the vertices that improve the solution by at least $\lceil \frac{\Delta}{x} \rceil$; b denotes the number of these vertices, i.e., $b = |B|$. We denote by y_1 the number of edges with one endpoint in B and the other in $V \setminus B$, and by y_2 the number of edges with both endpoints in B. By definition,

$$SOL \geq y_1 + y_2 = b\Delta - y_2 = \frac{b\Delta - y_1}{2} + y_1 = \frac{b\Delta + y_1}{2}. \tag{1}$$

We shall handle two cases, depending on the value of b with respect to k.

If $b < k$, then each vertex $v \in V \setminus B$ is not selected by ALGORITHM Mk VC-R(x) to be in B, because it is adjacent to at most $\lceil \frac{\Delta}{x} \rceil - 1$ vertices of $V \setminus B$. Thus, there are at least $\Delta - \lceil \frac{\Delta}{x} \rceil + 1$ edges that connect v with vertices in B. Summing up for all the vertices in $V \setminus B$, it holds that $y_1 \geq (n-b)\left(\Delta - \left\lceil \frac{\Delta}{x} \right\rceil + 1\right)$, and, considering also (1), we get

$$SOL \geq (n-b)\left(\Delta - \left\lceil \frac{\Delta}{x} \right\rceil + 1\right) + y_2 \tag{2}$$

$$SOL \geq \frac{b\Delta + (n-b)\left(\Delta - \left\lceil \frac{\Delta}{x} \right\rceil + 1\right)}{2}. \tag{3}$$

Using the upper bound for the optimum provided by Remark 1 and expressions (2) and (3), respectively, we get the following ratios:

$$\frac{SOL}{OPT} \geq \frac{(n-b)\left(\Delta - \left\lceil \frac{\Delta}{x} \right\rceil + 1\right) + y_2}{k\Delta} \geq \frac{(n-b)(x-1)}{kx} = \frac{n(x-1) - b(x-1)}{kx} \tag{4}$$

$$\frac{SOL}{OPT} \geq \frac{\frac{b\Delta + (n-b)\left(\Delta - \left\lceil \frac{\Delta}{x} \right\rceil + 1\right)}{2}}{k\Delta} \geq \frac{bx + (n-b)(x-1)}{2kx} = \frac{n(x-1) + b}{2kx}. \tag{5}$$

Observe that the right-hand side of (4) decreases with b while that of (5) increases; thus, the worst case occurs when the right-hand sides of them are equal, that is $\frac{n(x-1)-b(x-1)}{kx} = \frac{n(x-1)+b}{2kx} \Leftrightarrow b = \frac{n(x-1)}{2x-1}$, and hence

$$\frac{SOL}{OPT} \geq \frac{n(x-1) + \frac{n(x-1)}{2x-1}}{2kx} = \frac{n(x-1)}{k(2x-1)}. \tag{6}$$

If $b = k$, then trivially it holds that

$$\frac{SOL}{OPT} \geq \frac{k\left\lceil \frac{\Delta}{x} \right\rceil}{k\Delta} \geq \frac{1}{x}. \tag{7}$$

Note that (6) increases with x while (7) decreases; therefore, for the worst case, we have $\frac{n(x-1)}{k(2x-1)} = \frac{1}{x} \Leftrightarrow x = \frac{n+2k+\sqrt{4k^2+n^2}}{2n}$. In all, it holds that

$$\frac{SOL}{OPT} \geq \frac{2n}{n+2k+\sqrt{4k^2+n^2}}. \tag{8}$$

If $k < 0.55n$, the ratio of (8) leads to

$$\frac{SOL}{OPT} \geq \frac{2n}{n+2(0.55n)+\sqrt{4(0.55n)^2+n^2}} = \frac{2}{2.11+\sqrt{2.21}} = 0.55.$$

On the other hand, the ratio provided in Proposition 5 that holds for any algorithm, for $k > 0.55n$, gives $\frac{SOL}{OPT} \geq \frac{k}{n} \geq \frac{0.55n}{n} = 0.55$. □

Let us note that, as can be easily derived from (8), *when* $k = o(n)$, *the competitive ratio of* Algorithm MkVC-R *is asymptotical to* 1.

Algorithm MkVC-R(x) can be also used for the online max k-set coverage problem when all sets have the same cardinality. Nevertheless, this algorithm cannot give a competitive ratio better than $O\left(\frac{1}{\sqrt{k}}\right)$ for this case, which is much worse than the ratio achieved in Section 4. For completeness, its analysis is given in the Appendix.

3.2. Regular bipartite graphs

The max k-vertex coverage problem is NP-complete in bipartite graphs (by a reduction from the *densest k-subgraph* problem [6]). On the other hand, it is polynomial in regular bipartite graphs. In fact, if $k > n/2$, then an optimal solution that covers all the edges of the graph, i.e., $SOL = |E| = OPT$, can be obtained by selecting one of the color classes (since the graph is regular, each of them has size $n/2$) vertices of the bipartite graph (since the graph is regular, each of them has size $n/2$). If $k \leq n/2$ then, by selecting any k vertices from the same color class, we get a solution that covers $SOL = k\Delta$ which, by Remark 1, is optimal.

In this section, we present an improved competitive ratio for the max k-vertex coverage problem in regular bipartite graphs. A key point of such an improvement is that the maximum independent set can be found in polynomial time in bipartite graphs (see for example [13]). In what follows in this section, we consider that the number of vertices, n, is known *a priori*.

Our Algorithm MkVC-B initializes its solution with the first k released vertices. At this point, a maximum independent set B, of size $b \leq k$, in the graph induced by these k vertices is found. The vertices of this independent set will surely appear in the final solution. For the remaining $k-b$ vertices, we check if they cover at least $\frac{\frac{n\Delta}{2}-b\Delta}{\lceil\frac{n-b}{k-b}\rceil}$ edges different from those covered by the independent set B. If yes, we return the solution consisting of the b vertices of the independent set and these $k-b$ vertices. Otherwise, we wait for the next $k-b$ vertices and we repeat the test. In Algorithm MkVC-B, $G[A]$ denotes the subgraph of G induced by the vertex-subset A.

Algorithm MkVC-B

1: A ={the first k released vertices};
2: find a maximum independent set $B \subseteq A$ in $G[A]$;
3: set $b = |B|$;
4: **for** each released vertex v **do**
5: **if** $|A| = k$ **then**
6: **if** $m(A) \geq b\Delta + \frac{\frac{n\Delta}{2}-b\Delta}{\lceil\frac{n-b}{k-b}\rceil}$ **then**
7: **return** A;
8: **else**
9: $A = B$;
10: **else**
11: $A = A \cup \{v\}$;
12: **return** A;

Theorem 2. Algorithm MkVC-B *achieves a 0.6075-competitive ratio.*

Proof. Let us call a *batch* the set of the $k-b$ vertices of $A \setminus B$ in Lines 5–10 of Algorithm MkVC-B.

The solution computed by this algorithm contains a maximum independent set of size b. Since the input graph is bipartite, it holds that $b \geq \frac{k}{2}$.

The number of edges of the graph uncovered by the vertices of the maximum independent set is in total $\frac{n\Delta}{2} - b\Delta$. Any of these edges is covered by vertices belonging to at least one of the $\lceil \frac{n-b}{k-b} \rceil$ batches. Hence, on average, each batch covers $\frac{\frac{n\Delta}{2} - b\Delta}{\lceil \frac{n-b}{k-b} \rceil}$ of those edges; so there exists a batch that covers at least $\frac{\frac{n\Delta}{2} - b\Delta}{\lceil \frac{n-b}{k-b} \rceil}$ of them. Therefore, the algorithm covers in total at least $b\Delta + \frac{\frac{n\Delta}{2} - b\Delta}{\lceil \frac{n-b}{k-b} \rceil}$ edges. Using Remark 1, we get

$$\frac{SOL}{OPT} \geq \frac{b\Delta + \frac{\frac{n\Delta}{2} - b\Delta}{\lceil \frac{n-b}{k-b} \rceil}}{k\Delta} = \frac{b + \frac{\frac{n}{2} - b}{\lceil \frac{n-b}{k-b} \rceil}}{k}, \tag{9}$$

and since the last fraction in (9) increases with b, it holds that

$$\frac{SOL}{OPT} \geq \frac{\frac{k}{2} + \frac{\frac{n}{2} - \frac{k}{2}}{\lceil \frac{n - \frac{k}{2}}{k - \frac{k}{2}} \rceil}}{k} = \frac{k + \frac{n-k}{\lceil \frac{2n-k}{k} \rceil}}{2k}. \tag{10}$$

If $k \leq 0.6075n$, then (10) leads to $\frac{SOL}{OPT} \geq 0.6075$. Otherwise, using Proposition 5 we get the same ratio, and the theorem is concluded. □

Note that, by (10), ALGORITHM MkVC-B *achieves a competitive ratio asymptotical to* $\frac{3}{4}$ *when* $k = o(n)$.

3.3. Trees and chains

To the best of our knowledge, the complexity of the MAX k-VERTEX COVERAGE problem in trees is an open question. We prove here that it is polynomial by a transformation to the *quadratic* 0–1 *knapsack* (QUADRATIC KNAPSACK) problem. In the QUADRATIC KNAPSACK problem, we are given a set of variables V, a set E that expresses relations between the variables (quadratic relations), and an integer b. Each variable $i \in V$ has a cost c_i and a weight w_i associated with it, while each relation $(i, j) \in E$ has a cost c_{ij}. The goal is to find a subset A of V such that the total weight of A, i.e., $\sum_{i \in A} w_i$, does not exceed b and the total cost of A, i.e., $\sum_{i \in A} c_i + \sum_{i,j \in A} c_{ij}$, is maximized. Clearly, the instance of QUADRATIC KNAPSACK implies an underlying graph $G = (V, E)$. In the case where the underlying graph is an *edge series-parallel* graph, a polynomial algorithm has been proposed in [14].

Proposition 6. *The* MAX k-VERTEX COVERAGE *problem is polynomial in trees.*

Proof. Consider the tree $T = (V, E)$ to be the input for the MAX k-VERTEX COVERAGE problem. We construct an instance of the QUADRATIC KNAPSACK problem as follows: each $v_i \in V$ is assigned a weight $a_i = 1$ and a cost $c_i = d(v_i)$, and each $(v_i, v_j) \in E$ is assigned a cost $c_{ij} = -1$. We seek a QUADRATIC KNAPSACK of capacity $b = k$.

A solution to such an instance of the QUADRATIC KNAPSACK problem can be found in $O(nk^2)$ time by the dynamic programming algorithm presented in [14], as any tree is an edge series-parallel graph. Moreover, the cost of this solution corresponds to the value of an optimal solution for the MAX k-VERTEX COVERAGE problem in tree T, since, by the definition of the cost functions, each edge e covered by the solution is counted exactly once, independently of whether only one or both endpoints of e appear in the solution. □

In what follows in this section, we give algorithms that further improve the competitive ratios for the MAX k-VERTEX COVERAGE problem in trees and chains. Dealing with trees, the following result holds.

Proposition 7. *The* MAX k-VERTEX COVERAGE *problem can be solved within* $\left(1 - \frac{k-1}{\Delta^*}\right)$*-competitive ratio in trees, where* Δ^* *is the sum of the* k *largest degrees in the tree. The ratio is tight.*

Proof. An upper bound for the optimum solution is $OPT \leq \Delta^*$, which is the case where k non-adjacent vertices of the largest degree are selected.

Consider the algorithm that selects the k vertices of the largest degrees, breaking ties in an arbitrary way. These k vertices cover Δ^* edges, some of them possibly twice. It is easy to see that the number of such edges is maximized when the subgraph induced by the k selected vertices is connected. In this case, there are $k - 1$ edges covered twice. Hence, the total number of covered edges is $\Delta^* - (k - 1)$, while at most Δ^* edges can be covered by any solution.

The tightness of this ratio occurs in a straightforward way from the above proof, by considering as input a tree which has all its internal vertices of the same degree $\Delta > 1$. Thus, the algorithm can arbitrarily select a solution consisting of one connected component, while the optimal contains k non-adjacent vertices. □

Note that, if the number of vertices of degree greater than 1 is $r < k$, then our algorithm finds an optimum solution using just r vertices, since the edges that are adjacent to the leaves are covered by their other endpoints.

Furthermore, in the case where all the internal vertices of the tree have the same degree Δ, the ratio provided by Proposition 7 becomes $\left(1 - \frac{k-1}{k\Delta}\right)$. This ratio is better than the ratio proved for regular bipartite graphs in Theorem 2 for any $\Delta \geq 3$, but it is worse for $\Delta = 2$, i.e., in the case where the input graph is a chain.

An improvement for the MAX k-VERTEX COVERAGE problem in chains follows. The main idea of the algorithm is to partition the solution, A, into two disjoint parts, whose size is dynamically adjusted: the set B of vertices that contribute two edges in $E(A)$ and the set C of vertices that contribute one edge in $E(A)$.

ALGORITHM MkVC-C

```
1: A = ∅; (the solution of the algorithm)
2: B = ∅; (the set of vertices that increase the solution by exactly two)
3: C = ∅; (the set of vertices that increase the solution by exactly one)
4: In any step A = B ∪ C;
5: for each released vertex v do
6:   if |B| < k and v adds two new edges to the solution then
7:     if |A| = k then
8:       delete an arbitrary vertex from C;
9:     B = B ∪ {v};
10:  else if |A| < k and v adds one new edge to the solution then
11:    C = C ∪ {v};
12:    if the inclusion of v in A has as a result three consecutive vertices to appear in A then
13:      move v from C to B;
14:      remove the middle vertex from A;
15: return A;
```

Proposition 8. *For the* MAX k-VERTEX COVERAGE *problem in chains,* ALGORITHM *MkVC-C*

- *returns the (offline) optimum if* $k < \left\lceil \frac{n}{3} \right\rceil$ *or* $k \geq \left\lceil \frac{2n}{3} \right\rceil$, *and*
- *achieves a 0.75-competitive ratio if* $\left\lceil \frac{n}{3} \right\rceil \leq k < \left\lceil \frac{2n}{3} \right\rceil$.

Proof. Since there do not exist three consecutive vertices in the solution obtained by the algorithm, and taking into account that in C the two endpoints of the chain may appear, it holds that each vertex in B has at most one adjacent vertex in C, and hence $|B| \geq |C| - 2$.

If $k < \lceil \frac{n}{3} \rceil$, then assume, for contradiction, that in the final solution it holds that $C \neq \emptyset$, and let $v \in C$. Then

$$SOL = 2|B| + |C| \leq 2(k-1) + 1 = 2k - 1 < 2\left\lceil \frac{n}{3} \right\rceil - 1.$$

Thus, the non-covered edges of the input graph are at least $n - 2\left\lceil \frac{n}{3} \right\rceil + 2$. Since the number of connected components in the solution of the algorithm is at most $k - 1 < \left\lceil \frac{n}{3} \right\rceil - 1$, there are two adjacent edges not covered by the algorithm; let u be the common vertex of these edges. But the algorithm in Lines 6–9 should have removed v and added u in A, a contradiction. Thus, $C = \emptyset$, which means that all the k vertices of the solution cover privately exactly two edges, as in the optimum.

If $k \geq \left\lceil \frac{2n}{3} \right\rceil$, then the algorithm returns a solution covering all the edges of the graph. Indeed, since $|B| \geq |C| - 2$, we have

$$SOL = 2|B| + |C| = k + |B| \geq \left\lceil \frac{2n}{3} \right\rceil + \left(\left\lfloor \frac{n}{3} \right\rfloor - 1 \right) = n - 1 = |E|.$$

Hence, the optimum solution is obtained by the algorithm.

If $\left\lceil \frac{n}{3} \right\rceil \leq k < \left\lceil \frac{2n}{3} \right\rceil$, then for the solution created by the algorithm we have $SOL = 2|B| + |C|$, while $OPT \leq 2k = 2|B| + 2|C|$. Since $|B| \geq |C| - 2$, we get

$$\frac{SOL}{OPT} \geq \frac{2|B| + |C|}{2|B| + 2|C|} \geq \frac{2|B| + |B| + 2}{2|B| + 2|B| + 4} = \frac{3|B| + 2}{4|B| + 4} \simeq 0.75,$$

which completes the proof. □

4. Maximum k-(set)-coverage

A natural idea to obtain a good solution for the online MAX k-SET COVERAGE problem is, each time a new set P is released, to include P (removing another appropriate set) only if this inclusion implies a significant increase to the value of the current

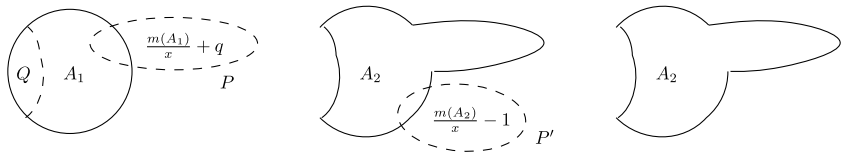


Fig. 2. An example of the execution of ALGORITHM Mk C.

solution. In [15], an algorithm that achieves a $\frac{1}{4}$-competitive ratio for the MAX k-SET COVERAGE problem is presented based upon this idea. More specifically, this algorithm selects a set P for inclusion only if the number of private elements of P is at least twice the number of the private elements of the candidate set for deletion.

In this section, we propose ALGORITHM Mk C, which uses the same natural idea but in a more general way. Our algorithm updates the current solution A_j only if, for some suitably selected set Q from A_j, the solution obtained after replacing in A_j the set Q by a candidate set P covers at least $m(A_j)\left(1+\frac{1}{k}\right)$ elements. Using this more general algorithm, we are able to prove that the competitive ratio is strictly greater than $\frac{1}{4}$, tending to $\frac{1}{4}$ as k increases. Our analysis is tight, and leads to better results for moderately large values of k than the analysis of the algorithm proposed in [15].

ALGORITHM Mk C

1: $j = 1$;
2: $A_j = \{$the first k released sets$\}$;
3: **for** each released set P **do**
4: find the set $Q \in A_j$ that covers privately the smallest number of elements in A_j;
5: **if** $m(A_j \setminus \{Q\} \cup \{P\}) > m(A_j) + \frac{m(A_j)}{k}$ **then**
6: $j = j + 1$;
7: $A_j = A_{j-1} \setminus \{Q\} \cup \{P\}$;

To analyze ALGORITHM Mk C, let A_z be the solution computed after having all the sets released, i.e., $SOL = m(A_z)$. Fix also an optimum solution A^*. We distinguish the following two types of *bad events* that may happen during the execution of the algorithm upon arrival of a set P:

(a) $P \in A^*$ and ALGORITHM Mk C does not select it, and
(b) $P \notin A^*$ and ALGORITHM Mk C discards $Q \in A^*$ in order to insert P into its current solution.

The total number of bad events of both types is a measure of the distance between A_z and A^*; clearly, at most k such events may happen. Notice also that more than one bad event of type (a) may happen while the current solution is kept unchanged, i.e. may correspond to some value of j in the algorithm, while at most one bad event of type (b) might correspond to it; so, let $\ell = |\{j$: some bad event of any type happens when the current solution is $A_j\}|$. Let A_{j_i}, $1 \le i \le \ell$, $1 \le j_i \le z$, be the ith of these current solutions, and let k_i, $1 \le i \le \ell$, be the number of events that occurred, with A_{j_i} being the current solution.

To make clear our notation as well as the execution of ALGORITHM Mk C, consider the example in Fig. 2. There, after the release of the first k sets, we get A_1 as the current solution. Assume that $Q \in A_1$ is the set of A_1 with the minimum number of private elements, and let q be the private elements of Q in A_1. Then, the set P which adds to the solution more than $\frac{m(A_1)}{x} + q$ more elements is released. The algorithm selects this set and removes Q. If $Q \in A^*$, then we have the first event; that is, $j_1 = 1$ and $A_{j_1} = A_1$. Next, the set P' which contains, say, $\frac{m(A_2)}{x} - 1$ private elements is released. Hence, the algorithm does not select it. If $P' \in A^*$, then we have the second event; that is, $j_2 = 2$ and $A_{j_2} = A_2$.

We will now provide an upper bound to $OPT = m(A^*)$ by some expression involving these A_{j_i}s, $1 \le i \le \ell$. Consider that the sth bad event corresponds to j_i, i.e., $\sum_{r=1}^{i-1} k_r < s \le \sum_{r=1}^{i} k_r$. Let P_s be the new set that arrives while the current solution is A_{j_i}, and let Q_s be the set that covers privately the smallest number of elements in A_{j_i}. Let, also, $\widetilde{Q}_s \subseteq Q_s$ be the set of private elements of Q_s in A_{j_i}.

If the event is of type (a), then $P_s \in A^*$ is not selected, and it covers a subset of the elements in $E(A_{j_i} \setminus \{Q_s\})$ plus its private elements, $\widetilde{P}_s \subseteq P_s$, in $E(A_{j_i} \setminus \{Q_s\} \cup \{P_s\})$. Note that it is $m(\widetilde{P}_s) \le m(\widetilde{Q}_s) + \frac{m(A_{j_i})}{k}$; otherwise P_s would be selected by the algorithm. Moreover, $m(\widetilde{Q}_s) \le \frac{m(A_{j_i})}{k}$, since Q_s has the smallest private part in A_{j_i}, and hence $m(\widetilde{P}_s) \le \frac{2m(A_{j_i})}{k}$.

In all, we get the following inclusion relation:

$$E(A^*) \subseteq \bigcup_{i=1}^{\ell} E(A_{j_i}) \cup \bigcup_{s} \widetilde{P}_s,$$

with s varying on the indices of (a)-type bad events (note that the sets of the optimum removed after a (b)-type bad event are always represented in the first term of the expression, since the union varies among all i for A_{j_i}s). This reduces trivially to

$$E(A^*) \subseteq E(A_{j_\ell}) \cup \bigcup_{i=2}^{\ell} \left[E(A_{j_{i-1}}) \setminus E(A_{j_i})\right] \cup \bigcup_{s} \widetilde{P}_s.$$

Thus, for the value of the optimum solution A^*, we have the following bound:

$$OPT \leq m(A_{j_\ell}) + \sum_{i=2}^{\ell} m(E(A_{j_{i-1}}) \setminus E(A_{j_i})) + \sum_{i=1}^{\ell} \left(k_i \frac{2m(A_{j_i})}{k} \right).$$

Lemma 1. $m(E(A_{j_{i-1}}) \setminus E(A_{j_i})) \leq \frac{|A_{j_{i-1}} \setminus A_{j_i}|}{k} m(A_{j_{i-1}})$, $2 \leq i \leq \ell$.

Proof. Let Q_r, $1 \leq r \leq |A_{j_{i-1}} \setminus A_{j_i}|$, be the rth set that is removed from $A_{j_{i-1}}$ between the events $i-1$ and i, considering only the sets that exist in $A_{j_{i-1}}$. Let, also, $\widetilde{Q}_r$ be the private part of Q_r just before it is removed. We will show that, for any p, $1 \leq p \leq |A_{j_{i-1}} \setminus A_{j_i}|$, it holds that $\sum_{r=1}^{p} q_r \leq \frac{p}{k} m(A_{j_{i-1}})$, and thus

$$m(E(A_{j_{i-1}}) \setminus E(A_{j_i})) = \sum_{r=1}^{\left|A_{j_{i-1}} \setminus A_{j_i}\right|} q_r \leq \frac{\left|A_{j_{i-1}} \setminus A_{j_i}\right|}{k} m\left(A_{j_{i-1}}\right).$$

Assume for a contradiction that for the first time after the removal of the set Q_p it holds that $\sum_{r=1}^{p} q_r > \frac{p}{k} m(A_{j_{i-1}})$; hence, $\sum_{r=1}^{p-1} q_r \leq \frac{p-1}{k} m(A_{j_{i-1}})$. Clearly, $q_p > \frac{m(A_{j_{i-1}})}{k}$. Moreover, according to ALGORITHM Mk C, Q_p has the smallest private part between the sets belonging in the solution when Q_p is selected to be removed. Thus, the $k-p$ sets of $A_{j_{i-1}}$ which are still in the solution have private parts of size greater than $(k-p)\frac{m(A_{j_{i-1}})}{k}$ in total. Consequently,

$$m(A_{j_{i-1}}) > \sum_{r=1}^{p} q_r + (k-p)\frac{m(A_{j_{i-1}})}{k} > \frac{p}{k} m(A_{j_{i-1}}) + (k-p)\frac{m(A_{j_{i-1}})}{k} = m(A_{j_{i-1}}),$$

a contradiction. Therefore, there is no p such that $\sum_{r=1}^{p} q_r > \frac{p}{k} m(A_{j_{i-1}})$, and the lemma is proved. □

Using Lemma 1, and since $m(A_{j_\ell}) > m(A_{j_i})$, $1 \leq i \leq \ell - 1$, and $\sum_{i=1}^{\ell} k_i = k$, we get

$$OPT \leq m(A_{j_\ell}) + \sum_{i=2}^{\ell} \frac{|A_{j_{i-1}} \setminus A_{j_i}|}{k} m(A_{j_{i-1}}) + \sum_{i=1}^{\ell-1} \frac{2m(A_{j_i})}{k} + (k-\ell+1)\frac{2m(A_{j_\ell})}{k}.$$

By definition, it holds that $j_\ell \leq z$, and hence $m(A_{j_\ell}) \leq m(A_z) = SOL$. Moreover, by ALGORITHM Mk C, $m(A_{j_\ell}) \geq \left(1+\frac{1}{k}\right)^{j_\ell - j_i} m(A_{j_i})$. Thus, for the ratio obtained by ALGORITHM Mk C, the following proposition holds.

Proposition 9. $\frac{SOL}{OPT} \leq \frac{1}{3+\frac{1}{k}\sum_{i=2}^{\ell} \frac{j_i - j_{i-1}+2}{\left(1+\frac{1}{k}\right)^{j_\ell - j_{i-1}}} - \frac{2(\ell-1)}{k}}$.

In what follows, we will give two bounds for the ratio given in Proposition 9. The first bound is not tight, but it definitely shows that the ratio is strictly better than $\frac{1}{4}$, while the second one is tight, but it is difficult to compute a close formula for it. For both bounds, the following lemma is used.

Lemma 2. *For any $\ell \geq 2$, it holds that* $\sum_{i=2}^{\ell} \frac{j_i - j_{i-1}+2}{\left(1+\frac{1}{k}\right)^{j_\ell - j_{i-1}}} \leq \frac{g(\ell)}{\ln\left(1+\frac{1}{k}\right)}$, *where* $g(\ell) = \frac{\left(1+\frac{1}{k}\right)^2}{e} \cdot e^{g(\ell-1)}$ *and* $g(2) = \frac{\left(1+\frac{1}{k}\right)^2}{e}$.

Proof. Set $d_i = j_i - j_{i-1}$. Consider the function $f_\ell(d) = \sum_{i=2}^{\ell} \frac{d_i+2}{\left(1+\frac{1}{k}\right)^{\sum_{j=i}^{\ell} d_j}}$. We will prove the lemma by induction to ℓ.

For $\ell = 2$, we have $f_2(d) = \sum_{i=2}^{2} \frac{d_i+2}{\left(1+\frac{1}{k}\right)^{\sum_{j=i}^{2} d_j}} = \frac{d_2+2}{\left(1+\frac{1}{k}\right)^{d_2}}$, where $\frac{\partial f_2(d)}{\partial d_2} = \frac{1-(d_2+2)\ln\left(1+\frac{1}{k}\right)}{\left(1+\frac{1}{k}\right)^{d_2}}$. The global maximum is attained for $d_2 + 2 = \frac{1}{\ln\left(1+\frac{1}{k}\right)}$. Thus, $f_2(d) \leq \frac{1}{\ln\left(1+\frac{1}{k}\right)} \cdot \frac{1}{\left(1+\frac{1}{k}\right)^{\frac{1}{\ln\left(1+\frac{1}{k}\right)}-2}} = \frac{1}{\ln\left(1+\frac{1}{k}\right)} \cdot \frac{\left(1+\frac{1}{k}\right)^2}{e}$.

Assume that the statement is true for $\ell - 1$. Then, we have, for ℓ,

$$f_\ell(d) = \sum_{i=2}^{\ell} \frac{d_i+2}{\left(1+\frac{1}{k}\right)^{\sum_{j=i}^{\ell} d_j}} = \frac{d_\ell+2}{\left(1+\frac{1}{k}\right)^{d_\ell}} + \sum_{i=2}^{\ell-1} \frac{d_i+2}{\left(1+\frac{1}{k}\right)^{\sum_{j=i}^{\ell} d_j}} = \frac{d_\ell+2}{\left(1+\frac{1}{k}\right)^{d_\ell}} + \frac{1}{\left(1+\frac{1}{k}\right)^{d_\ell}} \cdot f_{\ell-1}(d),$$

Table 1
Approximation ratio of ALGORITHM MkC.

k	2	3	5	10	30	50	100	300	500	1000
r	0.333	0.324	0.314	0.300	0.282	0.275	0.268	0.261	0.258	0.256

where $\frac{\partial f_\ell(d)}{\partial d_\ell} = \frac{1-(d_\ell+2+f_{\ell-1}(d))\ln\left(1+\frac{1}{k}\right)}{\left(1+\frac{1}{k}\right)^{d_\ell}}$. The global maximum is attained for $d_\ell + 2 + f_{\ell-1}(d) = \frac{1}{\ln\left(1+\frac{1}{k}\right)}$. Thus,

$$f_\ell(d) \le \frac{1}{\ln\left(1+\frac{1}{k}\right)} \cdot \frac{1}{\left(1+\frac{1}{k}\right)^{\frac{1}{\ln\left(1+\frac{1}{k}\right)}-2-f_{\ell-1}(d)}} \le \frac{1}{\ln\left(1+\frac{1}{k}\right)} \cdot \frac{\left(1+\frac{1}{k}\right)^2}{e} \cdot \left(1+\frac{1}{k}\right)^{\frac{g(\ell-1)}{\ln\left(1+\frac{1}{k}\right)}}$$
$$= \frac{1}{\ln\left(1+\frac{1}{k}\right)} \cdot \frac{\left(1+\frac{1}{k}\right)^2}{e} \cdot e^{g(\ell-1)},$$

and the lemma follows. □

To prove the first lower bound for the ratio achieved by ALGORITHM MkC, consider the following expression for the ratio, slightly coarser than the ratio of Proposition 9:

$$\frac{SOL}{OPT} \ge \frac{1}{3 + \frac{1}{k}\sum_{i=2}^{\ell} \frac{j_i - j_{i-1}}{\left(1+\frac{1}{k}\right)^{j_\ell - j_{i-1}}} + \frac{1}{k}\sum_{i=2}^{\ell}\left(\frac{2}{\left(1+\frac{1}{k}\right)^{j_\ell - j_{i-1}}} - 2\right)}. \tag{11}$$

Note that if $\ell = 1$ then both sums in the denominator become zero, and hence we have a $\frac{1}{3}$-competitive ratio. For $\ell \ge 2$, we may proceed to the following analysis. For the first sum in the denominator of (11), by a similar argument as in Lemma 2, we can prove that $\sum_{i=2}^{\ell} \frac{j_i - j_{i-1}}{\left(1+\frac{1}{k}\right)^{j_\ell - j_{i-1}}} \le \frac{g(\ell)}{\ln\left(1+\frac{1}{k}\right)}$, where $g(\ell) = \frac{1}{e} \cdot e^{g(\ell-1)}$ and $g(2) = \frac{1}{e}$. It is easy to see by simple induction that $g(\ell) \le 1$ for any $\ell \ge 2$, and hence

$$\sum_{i=2}^{\ell} \frac{j_i - j_{i-1}}{\left(1+\frac{1}{k}\right)^{j_\ell - j_{i-1}}} \le \frac{1}{\ln\left(1+\frac{1}{k}\right)} \le k. \tag{12}$$

For the second sum in the denominator of (11), we have

$$\sum_{i=2}^{\ell}\left(\frac{2}{\left(1+\frac{1}{k}\right)^{j_\ell - j_{i-1}}} - 2\right) \le \sum_{i=2}^{2}\left(\frac{2}{\left(1+\frac{1}{k}\right)} - 2\right) = \frac{2k}{k+1} - 2 = -\frac{2}{k+1}. \tag{13}$$

Therefore, using (12) and (13) in (11), we get the first lower bound claimed:

$$\frac{SOL}{OPT} \ge \frac{1}{4 - \frac{2}{k(k+1)}} = \frac{1}{4} + \frac{1}{4}\frac{1}{2k(k+1)-1}.$$

A tighter analysis can be obtained by using directly Lemma 2 and Proposition 9, and hence the following theorem gives the second lower bound on the ratio of ALGORITHM MkC.

Theorem 3. ALGORITHM MkC *achieves for the* MAX k-SET COVERAGE *problem a competitive ratio*

$$\frac{SOL}{OPT} \ge \frac{1}{3 + \frac{1}{k}\left[\frac{g(\ell)}{\ln\left(1+\frac{1}{k}\right)} - 2(\ell-1)\right]}$$

where $g(\ell) = \frac{\left(1+\frac{1}{k}\right)^2}{e} \cdot e^{g(\ell-1)}$ *and* $g(2) = \frac{\left(1+\frac{1}{k}\right)^2}{e}$.

The ratio of Theorem 3 is minimized for some $\ell = o(k)$. This ratio (denoted by r) for different values of k is shown in Table 1.

It is hopefully clear from the previous discussion that the analysis of ALGORITHM MkC works as well for the WEIGHTED MAX k-SET COVERAGE problem, up to the assumption that $m(\cdot)$ in ALGORITHM MkC denotes the total weight of the elements rather than their number.

We conclude this section by providing a tight example for the ratio achieved by ALGORITHM MkC. The idea of the example strongly relies upon the proof given above, which indicates the "critical" values of ℓ and j_i, $1 \le i \le \ell$. For simplicity, we will consider a case where $k = 3$, but it is easy to extend our example for any k, by appropriately choosing values for ℓ and j_i.

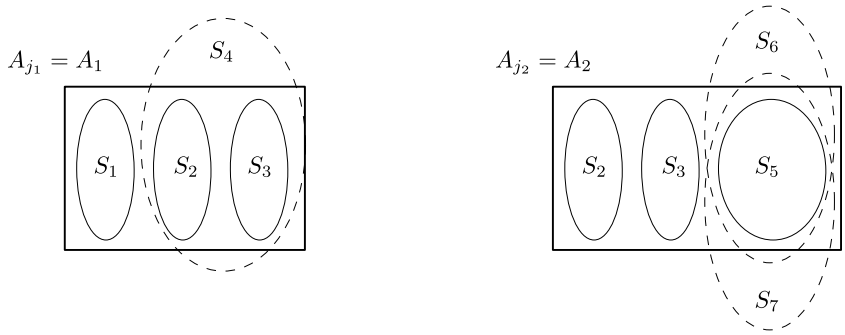


Fig. 3. A tight example for ALGORITHM MkC when $k = 3$.

Proposition 10. *The analysis of the ratio achieved by* ALGORITHM MkC *is tight.*

Proof. For $k = 3$, one can see that the ratio of ALGORITHM MkC is minimized when $\ell = 2$ and $j_2 - j_1 = 1$. Hence, consider the scenario shown in Fig. 3.

Let $A_1 = \{S_1, S_2, S_3\}$ be the solution after the first three sets have been released. These sets are disjoint, and each one covers c elements. Next, the set S_4 appears, which covers $2c - \epsilon$ new elements plus all elements in S_2 and S_3. The algorithm does not select S_4, since it may choose S_1 as a candidate for swapping; in this case, the new solution would cover $m(\{S_2, S_3, S_4\}) = 4c - \epsilon$ elements, which is smaller than $m(A_1) + \frac{m(A_1)}{k} = 4c$. Then, the set S_5 is released, which is disjoint to the previous sets and covers $2c$ elements. Thus, the algorithm replaces S_1 by S_5, and the new solution is $A_2 = \{S_2, S_3, S_5\}$. Finally, S_6 and S_7 are released, each one covering the elements in S_5 plus $\frac{8c}{3} - \epsilon$ new elements. ALGORITHM MkC does not select any of them, since they do not satisfy the algorithm's criterion.

So, the final solution, A_2, covers $m(S_2) + m(S_3) + m(S_5) = 4c$ elements. The optimal solution consists of sets S_4, S_6, and S_7 and covers

$$OPT = (4c - \epsilon) + \left(2c + \frac{8c}{3} - \epsilon\right) + \left(\frac{8c}{3} - \epsilon\right) = \frac{34c}{3} - \epsilon$$

elements. Therefore, the ratio achieved by ALGORITHM MkC is

$$\frac{SOL}{OPT} = \frac{4c}{\frac{34c}{3} - \epsilon} \simeq \frac{12}{34} = 0.353,$$

and the proof is completed. □

Note that the gap between this ratio and the ratio 0.324 given in Table 1 is due to the fact that the elements of S_1 do not appear in the optimal solution. Indeed, if S_1 is included in the optimal, then $OPT = \frac{37c}{3} - \epsilon$, and

$$\frac{SOL}{OPT} = \frac{4c}{\frac{37c}{3} - \epsilon} \simeq \frac{12}{37} = 0.324.$$

This gap decreases as $k \to \infty$.

5. Conclusions

There exist several interesting questions arising from the results presented in this paper. The first of them is to improve the easy $\frac{1}{2}$-competitive ratio for MAX k-VERTEX COVERAGE in general graphs and the (less easy) worst-case $\frac{1}{4}$-competitive ratio in set systems. Another open question is to provide tighter upper bounds for the online model handled in regular graphs. We still do not see how one can improve the analysis of ALGORITHM MkC in the case of equal cardinalities, or how to tighten the upper bound of Proposition 3 in Section 2, in order to match (or to get closer to) the competitive ratio of ALGORITHM MkC. Finally, especially in the case of MAX k-VERTEX COVERAGE, it would be interesting to tighten the negative results of Section 2 so that they fit the cases handled in Section 3, i.e., regular graphs and regular bipartite graphs. All the points raised in this section are subjects of ongoing research.

Acknowledgments

This research was supported by the French Agency for Research under the DEFIS program TODO, ANR-09-EMER-010. An extended abstract of this paper was presented at FCT 2011. We give many thanks to an anonymous referee for very useful comments and suggestions.

Appendix. Analysis of Algorithm MkVC-R for the max k-set coverage problem

We analyze, here, ALGORITHM MkVC-R(x) for the special case of the MAX k-SET COVERAGE problem in which every set contains exactly Δ elements. We prove that ALGORITHM MkVC-R(x) *provides an approximation ratio bounded below by* $\frac{2}{\sqrt{4k+1}+1} \backsim \frac{1}{\sqrt{k}}$, *and that this bound is asymptotically tight*.

Let $b = |B|$. Denote by A the solution computed by ALGORITHM MkVC-R, and fix an optimal solution A^*. As in the proof of Theorem 1, we distinguish two cases, with respect to the values of b and k.

If $b < k$, then consider the following partition of the optimum: $A^* = (A^* \cap A) \cup (A^* \setminus A)$. The first part can be bounded easily by $m(A^* \cap A) \leq m(A) = SOL$. For the second part, note that at most k sets contributing to it were not chosen by the algorithm. Thus, they represent individually an improvement of less than $\frac{\Delta}{x}$ with respect to A and, obviously, also with respect to $A^* \cap A$. So, the number of elements in the second part is at most $\frac{k\Delta}{x}$. The competitive ratio of ALGORITHM MkVC-R(x) can be written as

$$\frac{SOL}{OPT} \geq \frac{SOL}{SOL + k\frac{\Delta}{x}} = \frac{1}{1 + \frac{k\Delta}{x\,SOL}}$$

and, observing that $SOL \geq \Delta$, we obtain

$$\frac{SOL}{OPT} \geq \frac{1}{1 + \frac{k}{x}} = \frac{x}{x + k}. \tag{A.1}$$

If $b = k$, then, as in the proof of Theorem 1 (expression (7)), we have

$$\frac{SOL}{OPT} \geq \frac{1}{x}. \tag{A.2}$$

Since (A.1) increases with x while (A.2) decreases, some easy algebra shows that the optimal value for x is $\frac{\sqrt{4k+1}+1}{2}$ and, putting this in (A.1) or (A.2), the claimed bound follows.

For the tightness of the ratio, consider the following scenario. Assume that, initially, k sets that cover exactly the same Δ elements are released; the algorithm selects all of them. Then, $k-1$ sets are released; each of them covers $\frac{\Delta}{x} - 1$ elements privately. Thus, the algorithm does not select any of them, and hence $SOL = \Delta$. The optimal solution consists of one of the first sets plus the $k-1$ last sets, and hence $OPT = \Delta + (k-1)\left(\frac{\Delta}{x} - 1\right)$. Therefore, $\frac{SOL}{OPT} \simeq \frac{x}{x+k}$, which is the same ratio as when $b < k$ (due to (A.1)).

References

[1] A. Ageev, M. Sviridenko, Approximation algorithms for maximum coverage and max cut with given sizes of parts, in: Proceedings of the 7th International Conference on Integer Programming and Combinatorial Optimization, IPCO, in: LNCS, vol. 1610, Springer, 1999, pp. 17–30.
[2] N. Alon, B. Awerbuch, Y. Azar, N. Buchbinder, J. Naor, The online set cover problem, SIAM Journal on Computing 39 (2009) 361–370.
[3] G. Ausiello, N. Bourgeois, A. Giannakos, V.T. Paschos, Greedy algorithms for on-line set-covering, Algorithmic Operations Research 4 (2009) 36–48.
[4] B. Awerbuch, Y. Azar, A. Fiat, T. Leighton, Making commitments in the face of uncertainty: how to pick a winner almost every time (extended abstract), in: Proceedings of the 28th Annual ACM Symposium on Theory of Computing, STOC, ACM, 1996, pp. 519–530.
[5] N. Buchbinder, J. Naor, Online primal–dual algorithms for covering and packing, Mathematics of Operations Research 34 (2009) 270–286.
[6] F. Della Croce, V.T. Paschos, On the MAXk-VERTEX COVER problem, Tech. Rep. 307, Université Paris-Dauphine, cahiers du LAMSADE, March 2011.
[7] U. Feige, A threshold of ln n for approximating set cover, Journal of the ACM 45 (1998) 634–652.
[8] U. Feige, M. Langberg, Approximation algorithms for maximization problems arising in graph partitioning, Journal of Algorithms 41 (2001) 174–211.
[9] M.R. Garey, D.S. Johnson, L.J. Stockmeyer, Some simplified NP-complete graph problems, Theoretical Computer Science 1 (1976) 237–267.
[10] F. Grandoni, A. Gupta, S. Leonardi, P. Miettinen, P. Sankowski, M. Singh, Set covering with our eyes closed, in: Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science, FOCS, 2008, pp. 347–356.
[11] Q. Han, Y. Ye, H. Zhang, J. Zhang, On approximation of max-vertex-cover, European Journal of Operational Research 143 (2002) 342–355.
[12] D.S. Hochbaum, A. Pathria, Analysis of the greedy approach in problems of maximum k-coverage, Naval Research Logistics 45 (1998) 615–627.
[13] V.T. Paschos, A survey of approximately optimal solutions to some covering and packing problems, ACM Computing Surveys 29 (1997) 171–209.
[14] D.J. Rader Jr., G.J. Woeginger, The quadratic 0–1 knapsack problem with series-parallel support, Operations Research Letters 30 (2002) 159–166.
[15] B. Saha, L. Getoor, On maximum coverage in the streaming model & application to multi-topic blog-watch, in: Proceedings of the 9th SIAM International Conference on Data Mining, SDM, SIAM, 2009, pp. 697–708.